PRIMER

Chapter 4: Fundamentals of Digital Audio[1]

4.1. Sound Waves

In their very first introductions to physical science, students are taught that "sound is a wave." But what does this really mean, and what repercussions does this have in the way sound is represented in a computer? If you understand how sound is produced and transmitted, it will be easier for you to understand how it is digitally represented and manipulated, so let's take a closer look.

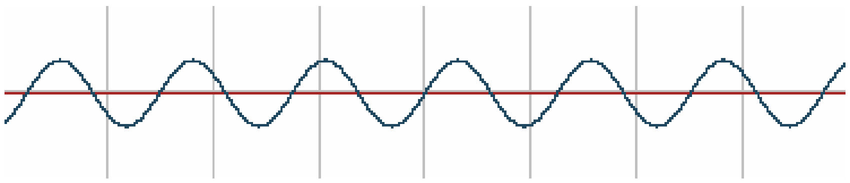


Figure 4.1. Pure Audio Tone No Overtones, Represented as a Wave Form

If you try to picture sound as a wave, you might picture a "bump" of air molecules moving across space. You might imagine the molecules moving up and down they as the sound wave makes its way to your ear. It makes a charming picture, but in fact, this isn't what a sound wave is at all.

Let's start at the beginning. First, sound is a *mechanical wave*, which means that it results from the motion of particles through a transmission medium – for example, in the case of sound, the motion of molecules in air. Because sound is a mechanical wave, it has to have something to move through; sound cannot be transmitted through a vacuum.

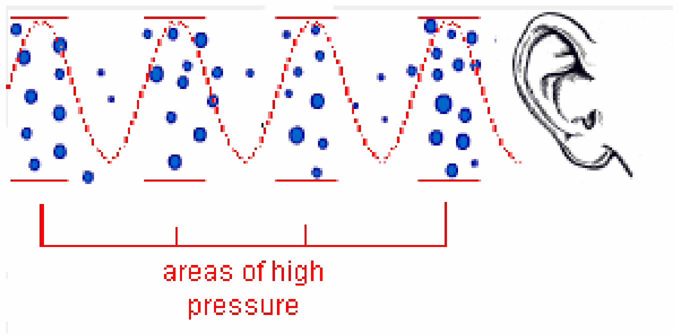


Figure 4.2. Changing Air Pressure Caused by Vibration of Air Molecules

The movement associated with a sound wave is initiated by a vibration. Imagine one of the strings inside a piano vibrating after one of the piano's soft hammers hits it. The air molecules next to the string are set in motion, radiating energy out from the vibrating string. For simplicity, let's just picture a single "wave" moving from left to right. As the string vibrates to the right, the molecules closest to the string are pushed to the right, bumping into the molecules next to them, which in a chain reaction bump into

[1] This material is based on work supported by the National Science Foundation under Grant No. DUE-0127280. This chapter was written by Dr. Jennifer Burg (burg@wfu.edu).

the molecules next to them, and so forth. When a group of molecules are pressed closer to their neighbors, air pressure rises. When the string moves back to the left, the molecules next to the string have space to spread out and move to the left as well, so the pressure between these molecules and the molecules to their right is reduced. This periodic changing of air pressure – high to low, high to low, etc. – radiates out from the string from left to right. (See Figure 4.2.)

So you see, a sound wave is, physically, not a bump of air moving across space. If you can visualize a sound wave as we just described it above, you can see that the motion of the air molecules is back and forth from left to right, the same direction in which the wave is radiating out from the string. A wave of this type is called a *longitudinal wave*, which is defined as a wave in which the motion of individual particles is in a direction parallel to the direction in which energy is being transported. Sound is a longitudinal mechanical wave.

Then why do we draw sound waves as we do, like the one in Figure 4.1? The sound wave in the figure is a graphical and mathematical abstraction of the physical phenomenon of sound. It represents the periodic change of air pressure. First the pressure increases as molecules are pushed closer together, shown by the upward bump in the graph. Then the pressure decreases as the molecules move apart, shown by the downward bump. These changes happen over time, so the x-axis in the graph represents time, while the y-axis represents air pressure.

A little terminology will make it easier for us to talk about sound waves in their graphical representation. A wave is said to be *periodic* if it repeats a pattern over time. The pattern that is repeated constitutes one *cycle* of the wave. A wavelength is the length (in distance) of one complete cycle. The *frequency* of a wave is the number of times a cycle repeats per unit time (which in the case of sound corresponds to the rate at which the air molecules are vibrating). Frequency is measured in cycles per second, or *herz* (abbreviated Hz). One cycle per second is one herz. One thousands cycles per second make 1000 herz, or one *kilohertz* (KHz). One million cycles per second is equal to 1000 KHz, which is one megahertz (MHz). The *period* of a wave is the amount of time it takes for one cycle to complete. Period and frequency are reciprocals of each other. That is,

$$period = 1/\ frequency$$

$$frequency = 1/\ period$$

The height of a wave is called its *amplitude*.

A graphical representation of sound in the form of a wave tells us something about the sound without our having to hear it. If the wave is completely regular like the one in Figure 4.1, then the sound is a pure tone, like a single musical note with no overtones. The amplitude of a wave corresponds to how loud the sound is; the larger the amplitude, the louder the sound. The frequency of a wave corresponds to the pitch of the sound; the higher the frequency, the higher-pitched the sound.

When sound is recorded, the changes in air pressure are translated to changes in voltage. A microphone picks up the changes in air pressure and records them as changes in voltage on an electrical wire. In the days of *analog* audio – 8-track tapes and vinyl record albums, for example – these voltage changes were captured in the form of changes in magnetic strength on the tape or changes in the depth of a groove on the vinyl record. A tape or record player then could play back the sound by reading the amplitude values imprinted on the tape or record, translating them again to voltages, and sending them to a speaker to be converted back to air vibrations.

When changes of air pressure reach the human ear in this wave-like pattern, they are detected by tiny hairs in the inner ear, translated into nerve impulses, sent to the brain, and in a miraculous process that is part of the human sensory system, interpreted as sound.

4.2. Adding and Decomposing Sound Waves

Figure 4.1 shows a simple wave form corresponding to a pure musical tone with no overtones. If you remember your high school trigonometry, you may have noticed that the shape of this wave is the same as the graph of a sine function. Few sounds in nature are this pure in form. For example, Figure 4.3 shows part of the wave produced by the spoken word "Hi!" It is not as regular as the pure tone of Figure 4.1, but it can be shown that it is, in fact, the sum of waves with a completely regular shape like the one in 4.1. (See Section 4.3 of the CS Chapter 4.)

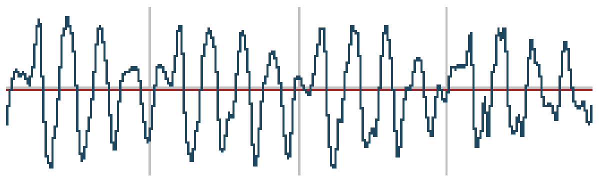


Figure 4.3. Part of the Spoken Word "Hi" Represented as a Wave Form

We can illustrate this with a simple example. Let's use musical tones that sound good together. Figures 4.4 a, b, and c shows the wave forms for three pure tones, the notes middle C, E, and G on the piano. Figure 4.4 d shows the wave form for notes C and E played simultaneously. Figure 4.4 e shows C, E, and G played simultaneously.

The importance of this fact is that, once we have represented a sound wave digitally, sound processing programs make it possible to analyze the wave form, filter out unwanted frequencies, and edit the sound for better quality or creative effects. Just as simple waves can be "added up," as shown below, a complex wave can be decomposed into its simple component parts. In fact, it can be proven that any periodic wave form, no matter how irregular it may appear, can be decomposed into a sum of pure sine and cosine waves. One of the mathematical methods to accomplish this decomposition is called the *Fourier transform*. In a digital sound processing program, you'll encounter the Fourier transform as the basis for filters that break down a sound and pull out unwanted frequencies, such as low-pitched noise. Understanding how wave forms are added and decomposed will help you understand the tools available to you in audio processing.

One last note about sound waves before we move on to digitization: You may have noticed that in our graphs of sound waves, we haven't included any units along the x- and y-axes. Time runs along the x-axis, and the units can sometimes be inferred from the example. In Figure 4.5a, for examples, which shows a tone that has a frequency of 400 Hz, ten complete cycles would cover 10/400th of a second. Generally, however, you don't need to consider either the time or the amplitude units in order to understand the example being presented. On the y-axis, representing amplitude, the unit of measurement could be *decibels*, a common measurement for the loudness of sound. It could also be voltage, since changes in voltage can be used by electronic devices to communicate changes in air pressure and thus the amplitude of sound. In these examples, the unit of measurement is not important; instead, we are focusing on the relative sizes of values on the y-axis and the precision with which they can be measured. (See Section 4.2 of CS Chapter 4 for a more precise definition of amplitude measured in decibels.)

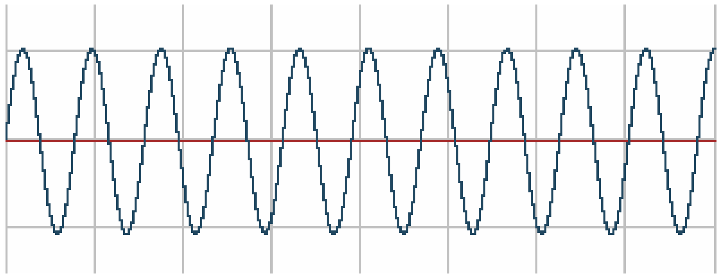


Figure 4.4a. The Musical Note C

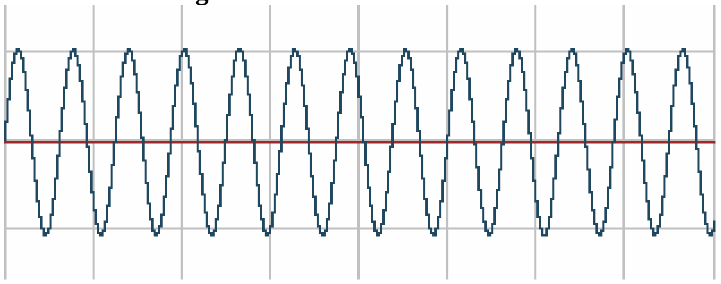


Figure 4.4b. The Musical Note E

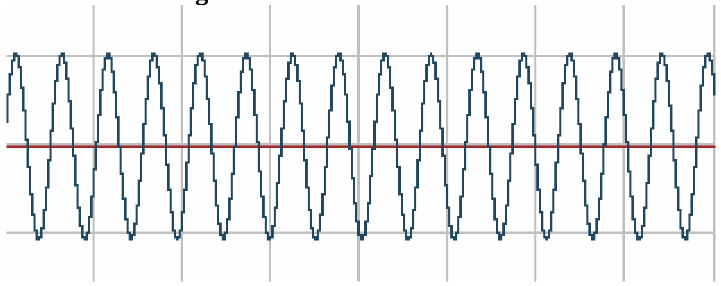


Figure 4.4c. The Musical Note G

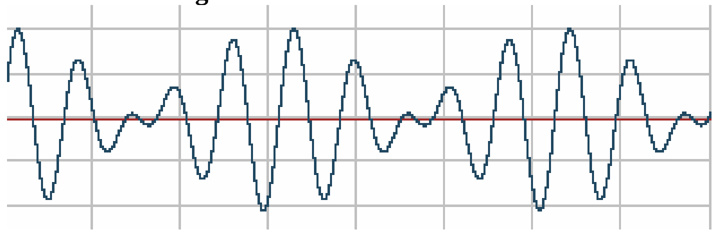


Figure 4.4d. The Notes C and E Played Together

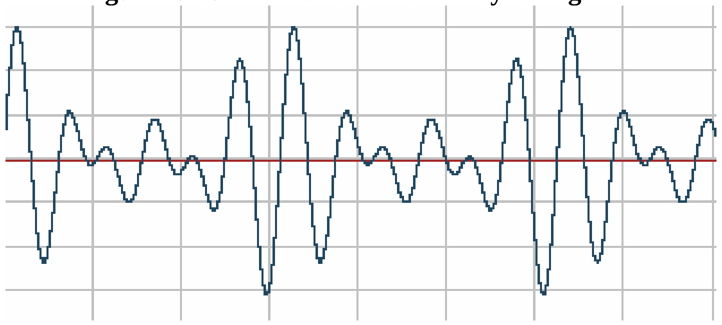


Figure 4.4e. The Notes C, E, and G Played Together

4.3. Digitizing Sound

We have seen that a wave form is a convenient way to describe sound, but it is an analog representation, not a digital one. A continuous, curved line like the wave in Figure 4.1 is in analog form in that, going horizontally along the x-axis, the line spans an infinite number of points in time. For any two points we might select on the graph, there is always another point in between. To reduce this infinite number of points to a finite number that a computer can handle, we must choose a number of equally-spaced points in time at which to *sample* the amplitude of the wave. Then each sample has to be quantized – that is, it must be represented in a fixed number of bits. Just like digitizing analog image data, digitizing analog audio data requires the two steps of sampling and quantization. In the domain of digital audio, this encoding process is often referred to as *pulse-code-modulation* (*PCM*).

The device that accomplishes the digitization process is called an *analog-to-digital converter* (*ADC*). Most up-to-date computers are equipped with sound cards that have an ADC to create digital audio. The microphone in the computer captures the sound and communicates it to the sound card, the sound card does the analog to digital conversion, and the data is stored in memory and/or on the hard disk. Let's look more closely at this digitization process, and consider the implications of converting from analog to digital form.

4.3.1. Sampling

In order to take samples of a sound wave, we need to choose a sampling rate. The choice of sampling rate will have an effect on how closely the digitized audio file matches the original sound wave. Like frequency, sampling rate is measured in Hz. A sampling rate of 44,100 samples per second is referred to as a sampling rate of 44,100 Hz, or 44.1 KHz. The question we want to explore is this: If you want to digitize a sound wave that has frequency of *n* Hz, what sampling rate is appropriate? The answer is that the sampling rate must be more than $2*n$ Hz, as we'll demonstrate below.

Figure 4.5a shows an analog sound wave with a frequency of 400 Hz. What happens if we try to digitize this wave by sampling it at a rate of 400 Hz – i.e., 400 samples per second? This would mean that we take just one sample for each cycle of the wave, at regularly-spaced intervals. Figure 4.5b shows the result. If we try to recreate the wave by joining the sample points, the wave we get is just a flat line.

In Figure 4.5c we try again, this time using a sampling rate of 600 Hz – i.e., 600 samples per second. That's two samples for every three cycles. Again, we aren't able to approximate the wave with so few samples. Even if we "round out" the wave that we get by joining the sample points, it isn't much like the original wave.

What happens if we try sampling a 400 Hz wave at 800 Hz? When we sample at exactly twice the frequency of the wave, we can reconstruct the original wave accurately if the samples are taken at the minimum and maximum amplitudes of the wave. But if we sample anywhere else, as in Figure 4.5d, we get back a wave with the right frequency but a lower amplitude. If we take the samples each time the wave crosses the x-axis, it's even worse – a flat wave of amplitude 0.

Sampling a 400 Hz wave at 1200 Hz, as in Figure 4.5e, definitely gives us enough information to reconstruct the wave – three samples every cycle. We have the correct amplitude and frequency, and we only need to round the wave out to get back the original.

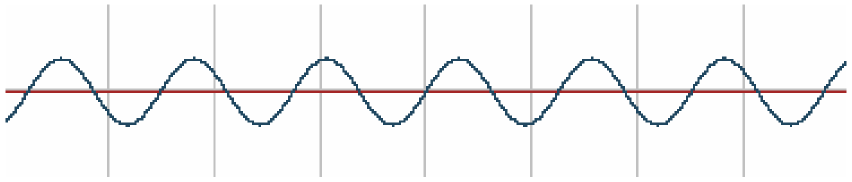


Figure 4.5a. A 400 Hz Wave Sampled at 44.1 MHz

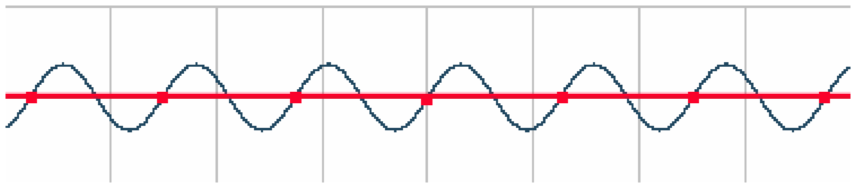


Figure 4.5b. A 400 Hz Wave Sampled at 400 Hz Becomes a Flat Wave

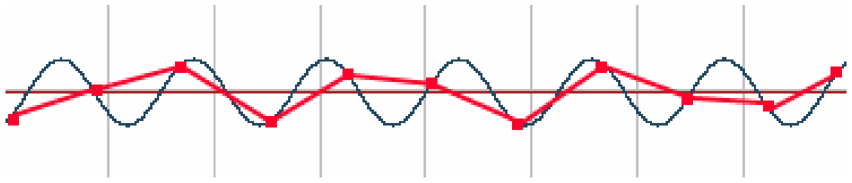


Figure 4.5c. A 400 Hz Wave Sampled at 600 Hz

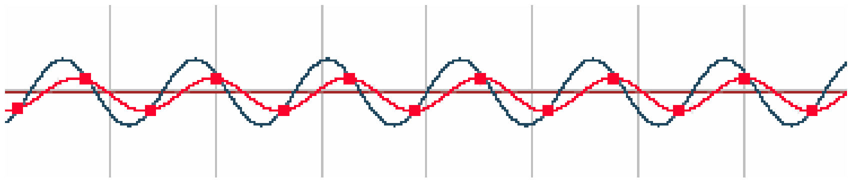


Figure 4.5d. A 400 Hz Wave Sampled at 800 Hz

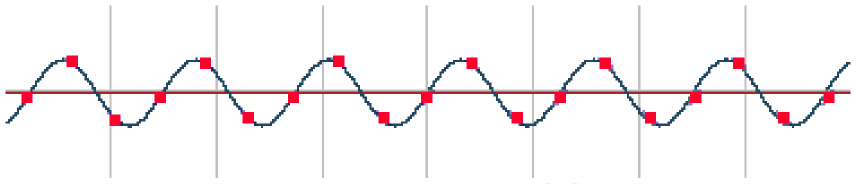


Figure 4.5e. A 400 Hz Wave Sampled at 1200 Hz

A Swedish scientist by the name of Nyquist was the first to observe formally what we've demonstrated here: To represent an analog periodic wave in digital form with the assurance that you can recreate it faithfully, you need to sample it more than twice in each cycle. To say this another way, the sampling rate must greater than twice the frequency of the wave, which is called the *Nyquist rate*. If the wave is of a complex form like the one shown in Figure 4.3, then the sampling rate must be greater than the frequency of the highest-frequency component. Sampling at a rate that is anything less than the Nyquist rate results in *sound aliasing* – that is, a frequency which “masquerades” as another because it is converted to digital form with a sampling rate that is not the same as the original.

We have looked at this in the case of a pure wave, but we can generalize the statement to more complex wave forms, which are sums of pure periodic waves. By the Nyquist theorem, if the highest frequency component of the sound being digitized is *n* Hz, then the sampling rate must be greater than $2*n$ Hz. The practical application of this is that when you record sound, you need to choose a sampling rate that is greater than twice the highest frequency that will be heard in the sound being recorded. For speech, 8000 samples per second generally suffices; for CD quality audio, 44,100 samples per second is the standard. If the sampling rate is too low, either you will have to put up with the “noise” inserted by aliased sound wave components, unless your sound editing software filters out the too-high frequency components before sampling.

4.3.2. Quantization

Quantization is the second step in analog-to-digital conversion. Once a sound wave has been sampled, each sample must be represented in a fixed number of bits. How do we know the proper number of bits to be used in each sample?

As explained in *Primer Chapter One*, the numbers that can be represented by n bits range in magnitude from 0 to $2^n - 1$, giving us 2^n different values. For example, with three bits we can represent $2^3 = 8$ values; with eight bits we can represent $2^8 = 256$ values; and with 16 bits we can represent $2^{16} = 65{,}536$ values.

Each sample of a sound wave has to be encoded in a fixed number of bits. The number of bits used in each sample is called the sound file’s *bit depth*. (The term *resolution* is used synonymously.) Let’s say we tell our sound processing program that we want to use 3-bit sound samples. (This isn’t a realistic example. The lowest bit-depth that is generally used for audio is eight bits per sample, but we want to keep our drawings simple.) A bit-depth of three bits would mean that the program would be able to represent only eight different amplitude levels. That’s like slicing the y-axis of a wave graph into eight equal segments, as shown in Figure 4.6. Each horizontal line in the figure represents one of the eight values that can be used to quantize the samples. For each sample, the amplitude value has to be rounded to the closest of these lines. This is shown for a number of discrete sample points in the graph. The dark blue squares are the sample points. The red square above or below each sample point is the value to which the point would be rounded when it is quantized.

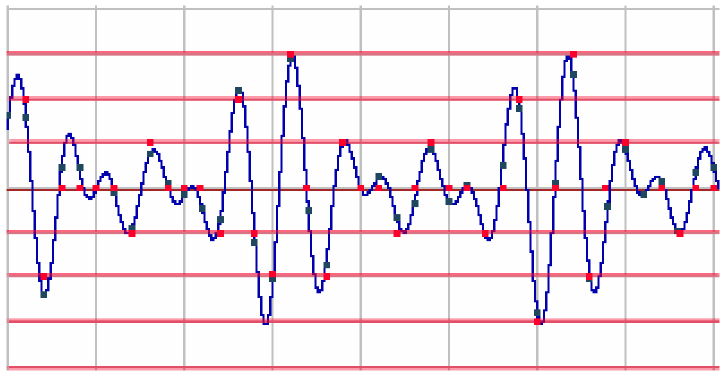


Figure 4.6. Quantization Error

You can see that quantization results in a loss of precision. Even if we smooth out the wave, we don't get back exactly the shape of the analog wave that captured the original sound. This loss of fidelity to the original that results from quantization error is called *distortion*.

The amount of distortion (also called *quantization noise*) can be pictured as another new component inserted into the original waveform. In Figure 4.7, the original waveform is shown in shadow, the quantized wave is in red, and the quantization error introduced is shown in green. Note that the original wave minus the error wave is equal to the quantized wave. This means that the quantization error introduces a low amplitude noise as another component added to the original waveform, which comes out sounding like a low amplitude hiss. The lower the amplitude of the original audio, the more distracting the noise because the amount of noise is large relative to the true sound.

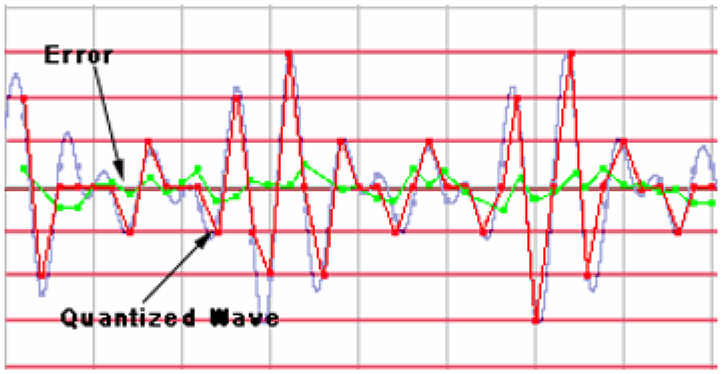


Figure 4.7. Original sound wave (in shadow), quantized wave (in red), and error wave (in green)

The point here is that the bit depth used for a sound file – the number of bits used for each sample – must be large enough for the audio quality desired. Some applications require more fidelity to the original sound source than others. For example, a digital recording of a symphony orchestra ought to sound as much like the live performance as possible, so a large bit depth is used – say 16 bits in each of two stereo channels. For telephone transmissions, on the other hand, it is important only that the voice be recognizable and understandable, so eight bits in a single channel is sufficient.

If a low bit depth is necessary but it introduces an unacceptable amount of distortion, a sound editing program can dither the sound file. *Dithering* is a process that removes some of the distortion caused by low bit depth by adding random noise to the audio file. The result is that instead of a distorted sound (like clicks or breaks), the audio file may have a little background hiss, but this noise is often preferable to distortion and closer to the original sound than without dithering. If you have to save an audio file at a lower bit depth than the original version, you should try the dithering option to see if the resulting quality is better.

4.3.3. Common Sampling Rates and Bit Depths

So why do you need to know about sampling rate and bit depth? First of all, any time you create a digital sound file, you'll be asked to choose a sampling rate and bit depth, and you'll have to select between stereo and mono. How do you know what to select?

The Nyquist theorem tells us that the sampling rate must be more than twice the frequency of the highest frequency component in the sound we want to digitize. The highest frequency that humans are capable of hearing is 20,000 Hz (i.e., 20 KHz). (That's a high upper limit. Most of us can't hear frequencies that high, and our ability to detect high frequency sounds diminishes as we age.) The standard sampling rate for audio CDs is 44.1 KHz. This is more than twice the frequency of the highest pitched tone humans are able to hear, so that's good enough. CD-quality digital audio uses 16 bit samples, and it is produced in stereo – two channels, that is. So CD-quality digital audio consists of two 16-bit samples taken 44,100 times for every second of sound. If you want to record music that is of CD-quality, you should choose a sampling rate of 44,100 KHz and a resolution of 16 bits in stereo.

If you're only recording the human voice (speaking), then you may not need the high fidelity of CD-quality sound. The highest frequency reached by the human voice is about 5 KHz, so a sampling rate of 10 KHz would be sufficient. A resolution of eight bits per sample and a sampling rate of 8 KHz may be enough (telephone quality digital audio) if you don't care about exact fidelity to the tones of the voice.

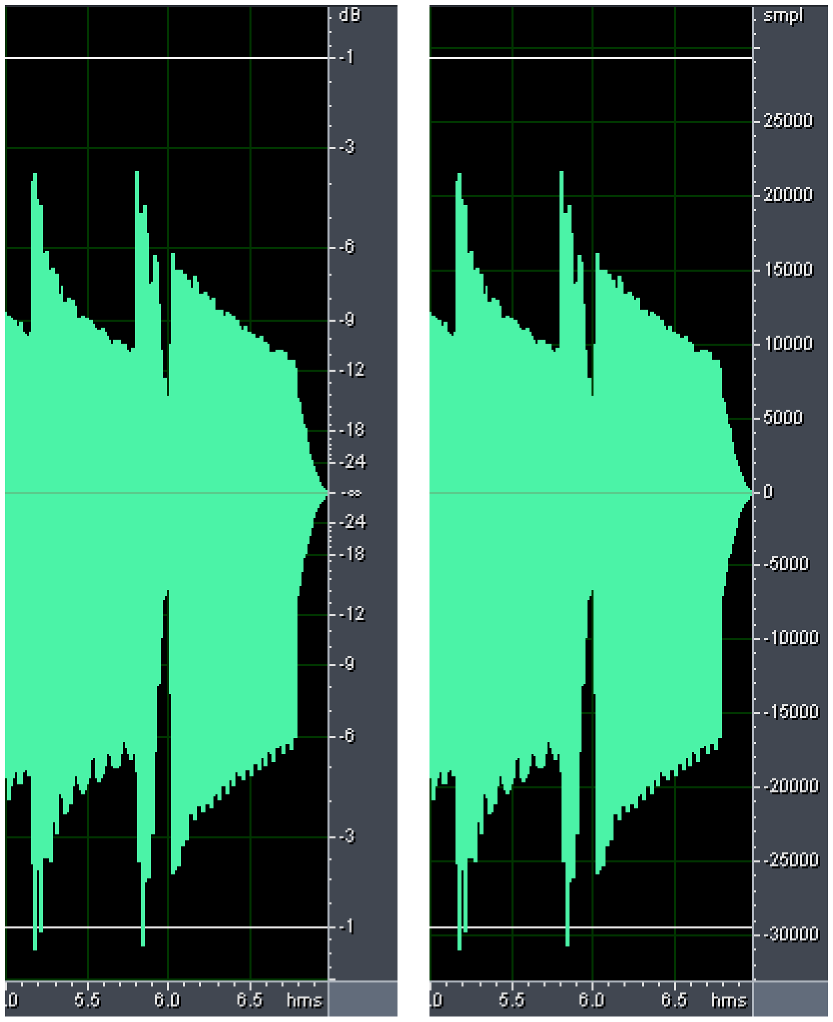
Table 4.2 lists common formats in terms of sampling rate, bit depth, and file size for 1 minute of digital audio.

4.4. Amplitude and Dynamic Range

The amplitude of a sound wave is an indication of the loudness of the sound, but the relationship between the two is not linear. In other words, one sound may have n times the amplitude of another, but this does not mean that it is perceived as n times louder to the human ear. For example, a voice at normal conversation level could be 100 times the air pressure amplitude of a soft whisper, but to human perception it seems only about 16 times louder. If we used units related to the changes in air pressure to measure sound, the differences in numbers wouldn't match the differences in the way we perceive sounds. Decibels are scaled to account for the non-linear nature of human sound perception. Table 4.1 gives the decibels of some common sounds (abbreviated dB).

Sound	**Decibels (dB)**
Threshold of hearing	0
Rustling leaves	20
Conversation	60
Jack hammer	100 (or more!)
Threshold of pain	140
Damage to eardrum	160

Table 4.1. Magnitude of common sounds measured in decibels



Amplitude in decibels **Amplitude in sample values**

Figure 4.8. Amplitude measured in decibels an samples values.

Decibels are an appropriate unit for describing the range of sound amplitudes, or dynamic range, of a digital audio file. In practice, the term *dynamic range* is used in two different contexts. A piece of music or an audio clip that ranges between very loud and very soft passages – for example, most classical symphonic music – is said to have a wide dynamic range. It is also possible to speak of the dynamic range achievable in an audio file based on the number of bits per sample in that file. If you open a new file to be recorded in a digital audio editing program and specify that you want 8 bits per sample, you are limiting the dynamic range of your digital recording to the dynamic range that 8 bits afford. The greater the bit depth, the greater the dynamic range. The importance of a greater dynamic range is that lower amplitude (softer) sounds are more affected by quantization noise when the dynamic range is small. That is, the lower the bit depth, the more the quiet parts of an audio file are distorted by quantization noise.

Often, audio editing programs allow you to change the view to show amplitude in sample values, percentages, or normalized values between 0 and 1. Figure 4.8 shows a 16-bit wave form in the decibel and sample-value views.

To summarize, the dynamic range of a digital audio file is limited by the bit depth, greater bit depth offering greater dynamic range. However, a particular piece of music may not use the full dynamic range offered by the bit depth in which it is captured. Some pieces of music use a wider dynamic range than others.

Dynamic range also comes into play in limiting the loudness of sounds that can be digitally recorded. An amplitude that exceeds a digital audio file's dynamic range will be clipped to the maximum level. Clipping causes extreme distortion of the audio signal, as evident in Figure 4.9 in the way the sound waves are "cut off" straight across the top. Digital audio editing programs have a level meter that shows the amplitude level and indicates when the amplitude is exceeding the dynamic range. Some also allow you to dynamically adjust the maximum amplitude level to avoid clipping.

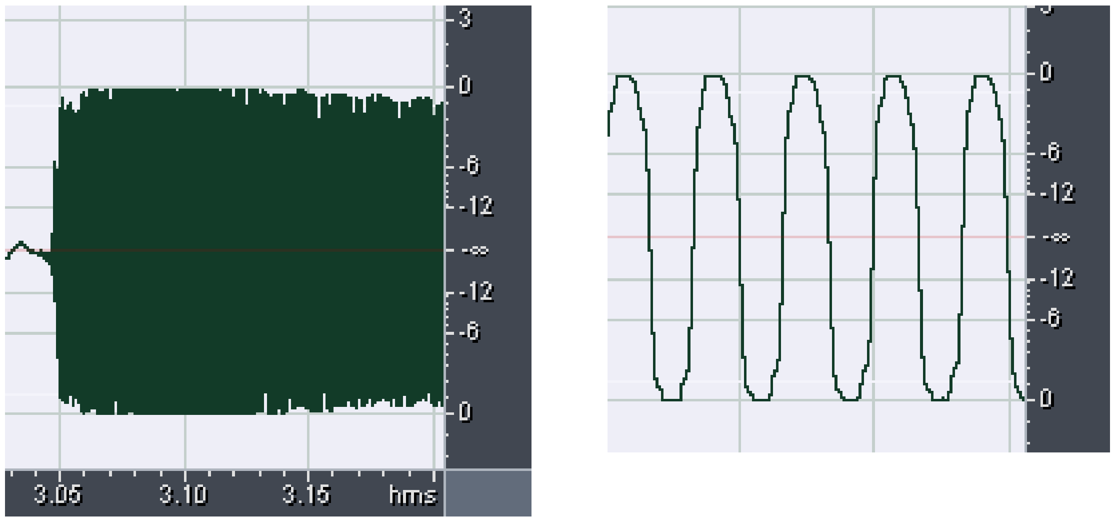


Figure 4.9. Clipped audio, zoomed out and close up

4.5. Digital Audio Files

4.5.1. File Sizes

Section 4.3 discussed the implications of your choice of sampling rate and bit depth, short-changing either one diminishes the fidelity of digital audio. If higher sampling rate and bit depth always give better quality sound, why not always choose the maximum possible? The problem is that you pay a price in the size of your digital audio files. Consider the size of a 60 minute CD that is recorded in stereo with a sampling rate of 44.1 KHz and a bit depth of 16 bits per sample. How large would this file be?

60 minutes * 60 seconds/minute = 3,600 seconds
3,600 seconds * 44,100 samples/second = 158,760,000 samples
one 16-bit value for each of the two stereo channels = 32 bits/sample
32 bits/sample * 1 byte/8 bits = 4 bytes/sample
158,760,000 samples * 4 bytes/sample = 635,040,000 bytes ≈ 630 MB

This is a large file. You'd fill up your hard disk drive pretty quickly with files of this size. You also have to consider file size if you're going to post your audio files on the web. The larger the file, the more time it takes to be downloaded to another person's computer. If memory space is limited or if you're creating audio files to be shared with others, you have to weigh the importance of audio quality against the size of the file.

Format	**Sampling Rate**	**Bit Depth**	**Uncompressed File Size in Bytes for One Minute of Audio**	**Download Time on 56Kb/s modem***	**Download Time on 1.5Mb/s cable modem**
speech (telephone)	8000 KHz	8 bits	480,000	1 minute 8 seconds	2.56 sec
CD stereo	44.1 KHz	16 bits per channel	5,292,000 (multiply by n for n-channel stereo)	> 25 minutes (assuming 2-channels)	> 9 min
DAT (digital audio tape)	48 KHz	16 bits per channel	5,760,000 (multiply by n for n-channel stereo)	> 27 minutes (assuming 2-channels)	> 10 min
DVD	96 KHz	24 bits per channel	17,280,000 (multiply by n for n-channel stereo)	> 82 minutes (assuming 2-channels)	> 30 min
*The values for 56K modem are underestimate, since in reality you don't get a full 56Kb/s.					

Table 4.2. Common sampling rates and bit depths for audio files

WORKSHEET
Link to Worksheet on Digital Audio File Size and Date Transfer Time

It is possible to retain audio quality to a great extent and still make your files smaller. This is done through audio compression. Compression techniques squeeze the data in an audio file into more concise formats so that important information is not lost. Many of the sound file types you are probably already familiar with use audio compression. When you work with a digital sound processing programming, you can choose the format that you want to save your audio file in. Common choices are listed in Table 4.3. Some of these choices automatically imply that the file will be compressed. Others allow you to indicate whether you want to compress the file, and possibly how much.

Your choice of file format should be determined by the type of sound you have recorded, your limits on file size, and who you expect will use the file. Some file formats work only on certain types of computers or operating systems. The differences are summarized in Table 4.3.

File Type	Acronym For	Originally Created By	Type of Compression	Platforms
.aiff	Audio Interchange File Format	Apple, adopted later by Silicon Graphics	usually not compressed, but has a compressed version	Apple Macintosh and Silicon Graphics computers, and now also on Windows
.wav		IBM and Microsoft	supports a number of different compression formats	primarily for Windows, but can be run on in other systems
.au and .snd	Also called mu-law or Sun mu-law format	Sun and NeXT	mu-law encoding compresses the file at a ratio of 2:1; slow decompression	Sun, NeXT, Unix or Linux operating system
.ra or .rm	Real Audio	Real	very high degree of compression; files can be streamed; sound quality poorer than .mp3	cross-platform
.mp3	MPEG audio layer 3	Moving Pictures Experts Group	good compression rate with high quality sound	cross-platform
.swa	Shockwave	Macromedia	uses same compression as mp3	cross- platform
.asf	advanced streaming format	Windows	proprietary compression algorithm	Primarily used with Windows Media Player

Table 4.3. Common Digital Audio File Formats

4.6. Audio Transforms and Filters

It is often convenient to change the representation of data in order to be able to separate out what is important and what is not important, or what you want to change and what you do not want to change. This is the case with digital audio data, where transforming the digital data from the time domain to the frequency domain makes it possible to handle different frequencies differently for the purposes of filtering and compression. Representing audio data in the digital domain entails recording amplitude values (the range) at discrete moments in time (the domain) as in Figure 4.10.

In the frequency domain, on the other hand, frequency values run along the x axis while amplitudes remain on the y axis. Accordingly, Figure 4.11 displays the amplitude of frequency component exists in a given audio clip. This clip has dominant frequency components of about 250 Hz, 2000 Hz, and 4800 Hz.

Spectral analysis provides another time-based way of viewing frequency information. In the spectral view, time is on the x-axis and frequency is on the y-axis. The amplitude of a each frequency at a moment in time is given by the color of that band across as you move horizontally. Bright colors correspond to higher amplitudes. In Figure 4.12, the highest amplitude frequencies are, as before, at approximately 250 Hz, 2000 Hz, and 4800 Hz. The amplitude remains constant through the interval shown.

Filters in audio editing programs use transforms (e.g., the Fast Fourier Transform, or FFT) to decompose a sound file into its frequency components, allowing the user to filter out frequencies in certain bands. An equalizer allows you to boost or reduce the amplitude in frequency bands. A graphic equalizer displays a graphical view of slider bars, each corresponding to a band of frequencies. You can move the sliders up and down depending on whether you want to make the frequency louder or softer. Specialized filters are also available in audio editing programs, including the de-esser, low pass, and high pass filter. A de-esser removes the hissing s sound that results when a person speaks or sings too close to a microphone. The low pass filter removes high frequencies above a given threshold, allowing low frequencies to pass through. The high pass filter does the opposite.

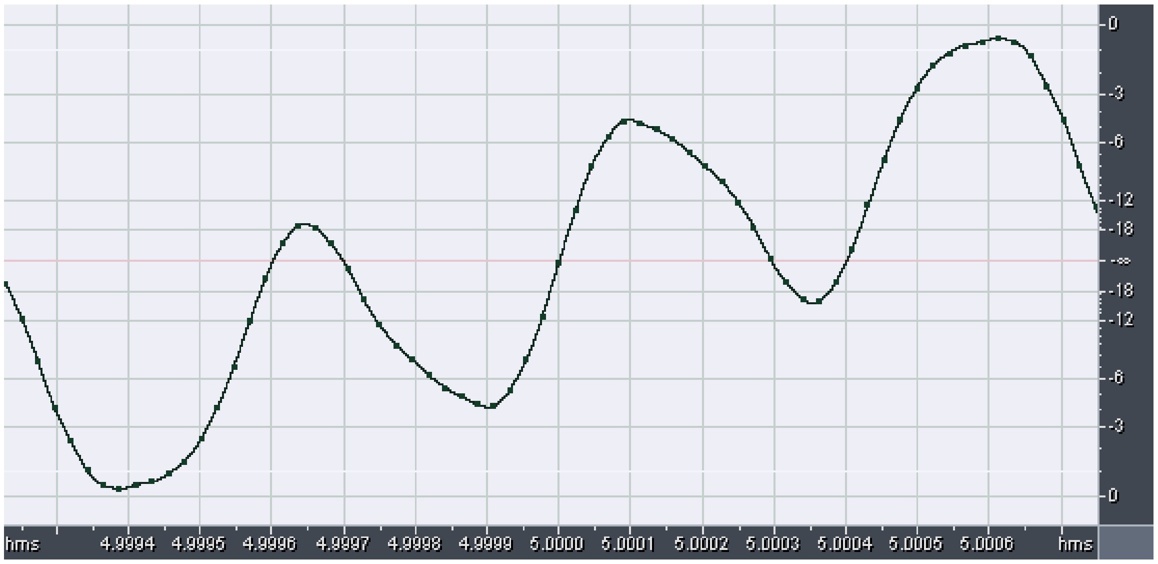


Figure 4.10. Digital audio in time domain (from the Waveform Edit View of Adobe Audition 1.0)

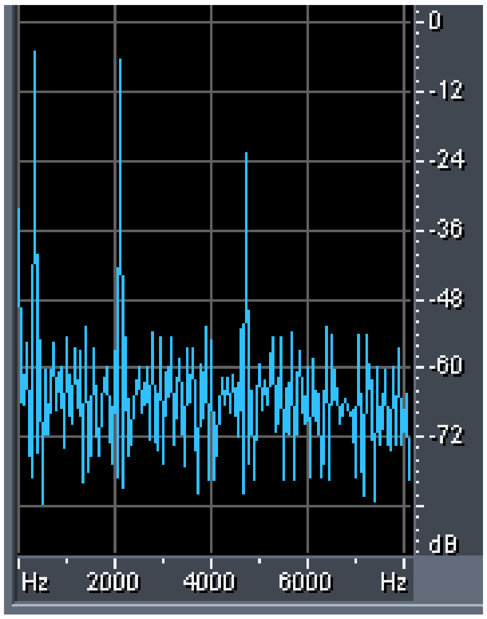


Figure 4.11. Digital audio in frequency domain (from the Frequency Analysis of Adobe Audition 1.0)

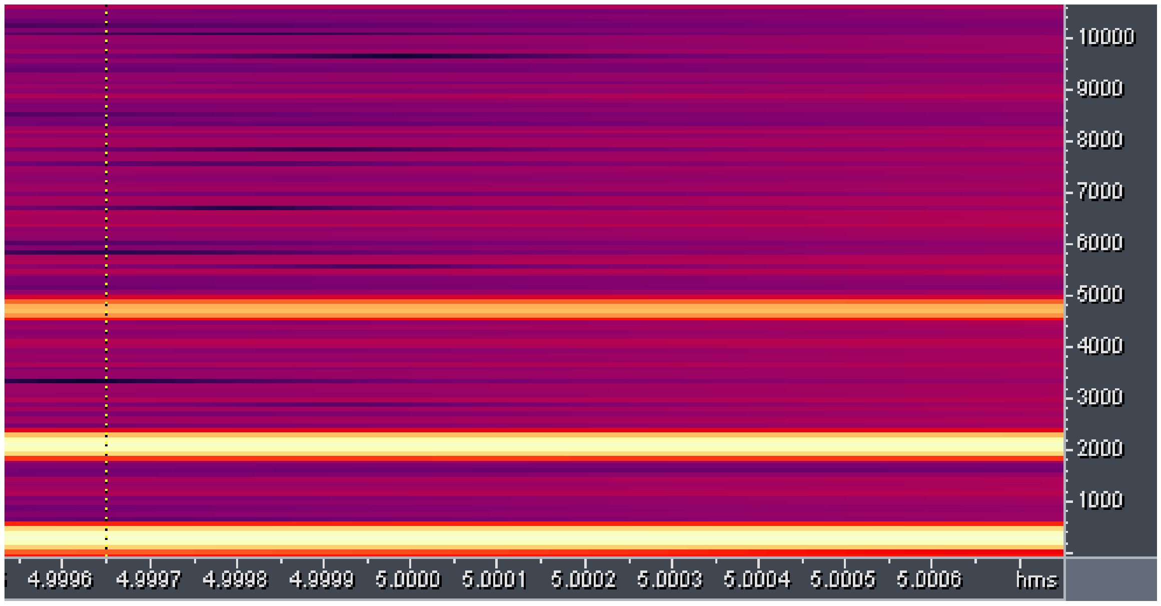


Figure 4.12. Spectral view of sound wave shown in Figure 4.10 (from Spectral View of Adobe Audition 1.0)

4.7. MIDI

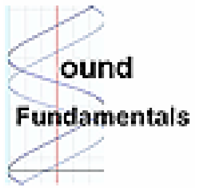
So far in this chapter, we have been describing digital audio. You have seen how digital audio is captured through samples of sound waves. Digital audio files consist of long sequences of samples, each of which is represented in a fixed number of bits. The digital-to-audio converter (DAC) attached to your computer's or stereo system's speakers converts these digital samples to voltages that tell the speakers how to vibrate and thus reproduce the original sound.

There is another way to store information about music, called the MIDI format. MIDI stands *for musical instrument digital interface*. MIDI defines a standard format in which electronic digital musical instruments can communicate with computers. A MIDI keyboard is an example of such a musical instrument. It looks like a small piano, but the difference is that instead of being a mechanical device that creates a sound by the striking of padded hammers on strings, it is an electronic device that synthesizes sound in digital form using its own internal microprocessor (i.e., computer).

MIDI music is recorded in a form completely different from the format of digital audio. While a piece of data in a digital audio file represents the amplitude of a sound wave at some point in time, a piece of data in a MIDI file represents the instrument being played, the note being played, and the duration of the note. As a piece of music is played into a MIDI instrument, the instrument's internal computer records the music in the MIDI format. Then the music can be passed to your desktop or laptop computer by means of a MIDI cable connection

MIDI files are compact and easy to work with. Once you get your MIDI file onto your computer, you can work with it in a MIDI music processing program, easily changing the key, tempo, or instrument being played.

If MIDI files are so compact and easy to work with, you may wonder why we don't use the MIDI format for all audio files. The reason is that MIDI works for simulating or reproducing the sound of musical instruments, but it doesn't work for recording real-time audio events. Also, MIDI music is essentially instrumental music. (It's true that the human voice can be synthesized, but we'd have to be able to synthesize every possible word a human could say.) Sometimes we want to record exactly what our ears hear, and for this we need digital audio recording.



Link to on-line demo on Digital Audio Fundamentals